# Red Hat build of OpenJDK 21

## 在 RHEL 上安装并使用红帽构建的 OpenJDK 21

Last Updated: 2024-05-10

## 法律通告



## 摘要

Red Hat build of OpenJDK 是 Red Hat Enterprise Linux 平台上的红帽产品。安装和使用红帽构建的 OpenJDK 21 指南概述了此产品，并解释了如何安装软件并开始使用它。

# 目录

# 提供有关红帽构建的 OPENJDK 文档的反馈

要报告错误或改进文档，请登录到 Red Hat JIRA 帐户并提交问题。如果您没有 Red Hat Jira 帐户，则会提示您创建一个帐户。

**流程**

1. 单击以下链接 **以创建 ticket**。
2. 在 **Summary** 中输入问题的简短描述。
3. 在 **Description** 中提供问题或功能增强的详细描述。包括一个指向文档中问题的 URL。
4. 点 **Submit** 创建问题，并将问题路由到适当的文档团队。

# 使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

# 第 1 章 红帽构建的 OPENJDK 21 概述

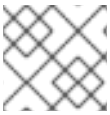
OpenJDK（Open Java Development Kit）是 Java Platform, Standard Edition (Java SE) 的一个开源实现。红帽构建的 OpenJDK 有四个版本： 8u、11u、17u 和 21u。

红帽构建的 OpenJDK 软件包在 Red Hat Enterprise Linux 和 Microsoft Windows 上提供，并作为红帽生态系统目录中的 JDK 和 JRE 提供。

# 第 2 章 在 RED HAT ENTERPRISE LINUX 上安装红帽构建的 OPENJDK 21

红帽构建的 OpenJDK 是用于开发和运行各种平台相关应用程序的环境，从移动应用程序到桌面和 Web 应用程序和企业系统。红帽提供了 Java Platform SE （标准版）的开源实现，称为红帽构建的 OpenJDK。

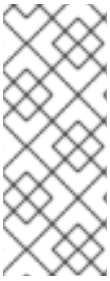
应用程序使用 JDK (Java Development Kit)开发。应用程序在 JVM (Java 虚拟机)上运行，该虚拟机包含在 JRE (Java Runtime Environment)和 JDK 中。还有一个无头版本的 Java，其占用空间最小，不包括用户界面所需的库。无头版本打包在无头子软件包中。



**注意**

如果您不确定是否需要 JRE 或 JDK，建议您安装 JDK。

以下小节提供了在 Red Hat Enterprise Linux 上安装红帽构建的 OpenJDK 的说明。



**注意**

您可以在本地系统中安装 Red Hat build of OpenJDK 的多个主版本。如果您需要从一个主版本切换到另一个主版本，请在命令行界面(CLI)中运行以下命令，然后按照屏幕提示操作：

```
$ sudo update-alternatives --config 'java'
```

## 2.1. 使用 YUM 在 RHEL 上安装 JRE

您可以使用系统软件包管理器 **yum** 安装红帽构建的 OpenJDK Java Runtime Environment (JRE)。

**先决条件**

- 在系统上以具有 root 权限的用户身份登录。
- 将本地系统注册到您的 Red Hat Subscription Manager 帐户。请参阅 *使用 Red Hat Subscription Manager 用户指南 注册系统*。

**流程**

1. 运行 **yum** 命令，指定您要安装的软件包：

```
$ sudo yum install java-21-openjdk
```

2. 检查安装是否正常工作：

```
$ java -version
```

**注意**

如果上一命令的输出显示您系统上签出了不同的 OpenJDK 主版本，您可以在 CLI 中输入以下命令来将您的系统切换为使用 Red Hat build of OpenJDK 21：

```
$ sudo update-alternatives --config 'java'
```

## 2.2. 使用存档在 RHEL 上安装 JRE

您可以使用存档安装红帽构建的 OpenJDK Java Runtime Environment (JRE)。如果 Java 管理员没有 root 特权，这非常有用。

**注意**

为便于升级，请创建一个父目录，使其包含您的 JRE，并使用通用路径创建指向最新 JRE 的符号链接。

**流程**

1. 创建一个目录，您要下载存档文件，然后导航到命令行界面(CLI)上的该目录。例如：

```
$ mkdir ~/jres

$ cd ~/jres
```

2. 导航到红帽客户门户网站中的 Software Downloads 页面。
3. 从 **Version** 下拉列表中选择最新版本的 OpenJDK 21，然后将 Linux 的 JRE 归档下载到本地系统。
4. 将存档的内容提取到您选择的目录中。
例如：

```
$ tar -xf archive_file_name.tar.gz -C ~/jres
```

5. 使用指向 JRE 的符号链接创建通用路径，以便更轻松地升级：

```
$ ln -s ~/jres/archive_file_name ~/jres/java-21
```

6. 配置 **JAVA_HOME** 环境变量：

```
$ export JAVA_HOME=~/jres/java-21
```

7. 验证 **JAVA_HOME** 环境变量是否已正确设置：

```
$ printenv | grep JAVA_HOME
JAVA_HOME=~/jres/java-21
```

**注意**

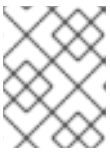
使用此方法安装时，Java 仅适用于当前用户。

8. 将通用 JRE 路径的 **bin** 目录添加到 **PATH** 环境变量中：

```
$ export PATH="$JAVA_HOME/bin:$PATH"
```

9. 验证 **java -version** 是否在没有提供完整路径的情况下工作：

```
$ java -version
```



**注意**

您可以通过在 ~/**.bashrc** 中导出环境变量来确保 **JAVA_HOME** 环境变量为当前用户保留。

## 2.3. 使用 YUM 在 RHEL 上安装红帽构建的 OPENJDK

您可以使用系统软件包管理器 **yum** 安装红帽构建的 OpenJDK。

**先决条件**

- 以具有 root 权限的用户身份登录。
- 将本地系统注册到您的 Red Hat Subscription Manager 帐户。请参阅 *使用 Red Hat Subscription Manager 用户指南 注册系统*。

**流程**

1. 运行 **yum** 命令，指定您要安装的软件包：

```
$ sudo yum install java-21-openjdk-devel
```

2. 检查安装是否正常工作：

```
$ javac -version

javac 21.0.1
```

## 2.4. 使用存档在 RHEL 上安装红帽构建的 OPENJDK

您可以使用存档安装红帽构建的 OpenJDK。如果 Java 管理员没有 root 特权，这非常有用。

**注意**

为简化升级，创建一个包含您的 JRE 的父目录，并使用通用路径创建指向最新 JRE 的符号链接。

**流程**

1. 创建一个目录，您要下载存档文件，然后导航到命令行界面(CLI)上的该目录。例如：

```
$ mkdir ~/jdks

$ cd ~/jdks
```

2. 导航到红帽客户门户网站中的 Software Downloads 页面。
3. 从 **Version** 下拉列表中选择最新版本的 OpenJDK 21，然后将 Linux 的 JDK 存档下载到本地系统。
4. 将存档的内容提取到您选择的目录中：

```
$ tar -xf archive_file_name.tar.xz -C ~/jdks
```

5. 使用到 JDK 的符号链接创建通用路径，以便更轻松地升级：

```
$ ln -s ~/jdks/archive_file_name ~/jdks/java-21
```

6. 配置 **JAVA_HOME** 环境变量：

```
$ export JAVA_HOME=~/jdks/java-21
```

7. 验证 **JAVA_HOME** 环境变量是否已正确设置：

```
$ printenv | grep JAVA_HOME
JAVA_HOME=~/jdks/java-21
```

**注意**

使用此方法安装时，Java 仅适用于当前用户。

8. 将通用 JRE 路径的 **bin** 目录添加到 **PATH** 环境变量中：

```
$ export PATH="$JAVA_HOME/bin:$PATH"
```

9. 验证 **java -version** 是否在没有提供完整路径的情况下工作：

```
$ java -version
```

**注意**

您可以通过在 **~/.bashrc** 中导出环境变量来确保 **JAVA_HOME** 环境变量为当前用户保留。

## 2.5. 使用 YUM 在 RHEL 上安装红帽构建的 OPENJDK 的多个主版本

您可以使用系统软件包管理器 **yum** 安装多个版本的 OpenJDK。

**先决条件**

- 一个具有有效订阅的 Red Hat Subscription Manager (RHSM)帐户，可访问提供您要安装的 OpenJDK 的红帽构建的仓库。
- 必须具有系统上的 root 权限。

**流程**

1. 运行以下 **yum** 命令以安装软件包：
对于红帽构建的 OpenJDK 21

```
$ sudo yum install java-21-openjdk
```

对于红帽构建的 OpenJDK 17

```
$ sudo yum install java-17-openjdk
```

对于红帽构建的 OpenJDK 11

```
$ sudo yum install java-11-openjdk
```

对于红帽构建的 OpenJDK 8

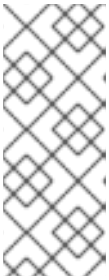
```
$ sudo yum install java-1.8.0-openjdk
```

2. 安装后，检查可用的 Java 版本：

```
$ sudo yum list installed "java*"
```

3. 检查当前的 java 版本：

```
$ java -version
```



**注意**

您可以在本地系统中安装 Red Hat build of OpenJDK 的多个主版本。如果您需要从一个主版本切换到另一个主版本，请在命令行界面(CLI)中运行以下命令，然后按照屏幕提示操作：

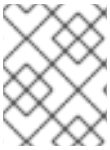
```
$ sudo update-alternatives --config 'java'
```

**其他资源**

- 您可以使用 **java --alternatives** 配置要使用的默认 Java 版本。如需更多信息，请参阅 在 RHEL 上非交互选择系统范围的红帽 OpenJDK 版本构建。

## 2.6. 使用存档在 RHEL 上安装多个 OPENJDK 主版本

*您可以使用 与使用存档在 RHEL 上安装 JRE 的相同步骤，或使用多个主版本在 RHEL 8 上安装 红帽构建的 OpenJDK 中的内容安装 Red Hat build of OpenJDK 的多个主版本。*



**注意**

有关如何为系统配置默认的 OpenJDK 版本红帽构建的步骤，请参阅 选择系统范围的 java 版本。

**其他资源**

- 有关安装 JRE 的说明，请参阅使用存档在 RHEL 上安装 JRE 。

- 有关安装 JDK 的说明，请参阅使用存档在 RHEL 8 上安装红帽构建的 OpenJDK 。

## 2.7. 使用 YUM 在 RHEL 上安装多个 OPENJDK 次版本

您可以在 RHEL 上安装 Red Hat build of OpenJDK 的多个次版本。这可以通过防止安装的次版本被更新。

**先决条件**

- 从非交互 中选择系统范围的红帽构建的 OpenJDK 版本，在 RHEL 上选择系统范围的 OpenJDK 版本。

**流程**

- 在 **/etc/yum.conf** 目录中添加 **installonlypkgs** 选项，以指定 **yum** 可安装的 OpenJDK 软件包的红帽构建，但不更新。

  ```
  installonlypkgs=java-<version>-openjdk,java-<version>-openjdk-headless,java-<version>-openjdk-devel
  ```

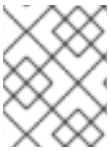
  更新将在系统中保留旧版本时安装新软件包。

**注意**

Red Hat build of OpenJDK 的不同次版本可在 **/usr/lib/jvm/ *<minor version>*** 文件中找到。

## 2.8. 使用存档在 RHEL 上安装多个 OPENJDK 次版本

安装多个次版本与使用存档在 RHEL 上安装 JRE，或使用使用多个次版本的归档在 RHEL 8 上安装红帽构建的 OpenJDK 相同。



**注意**

有关如何为系统选择默认次版本的说明，请参阅 在 RHEL 上非交互选择系统范围的 Red Hat build of OpenJDK 版本。

**其他资源**

- 有关安装 JRE 的说明，请参阅使用存档在 RHEL 上安装 JRE 。
- 有关安装 JDK 的说明，请参阅使用存档在 RHEL 8 上安装红帽构建的 OpenJDK 。

# 第 3 章 红帽构建的 OPENJDK 21 的调试符号

调试符号有助于调查红帽构建的 OpenJDK 应用程序崩溃。

## 3.1. 安装调试符号

这个步骤描述了如何为红帽构建的 OpenJDK 安装调试符号。

**先决条件**

- 在本地 sytem 上安装了 **gdb** 软件包。
  - 您可以在 CLI 上发出 **sudo yum install gdb** 命令，以在本地系统上安装此软件包。

**流程**

1. 要安装调试符号，请输入以下命令：

```
$ sudo debuginfo-install java-21-openjdk

$ sudo debuginfo-install java-21-openjdk-headless
```

这些命令安装 **java-21-openjdk-debuginfo**、**java-21-openjdk-headless-debuginfo** 以及为红帽构建的 OpenJDK 21 二进制文件提供调试符号的额外软件包。这些软件包并不是自我的，*不包含*可执行二进制文件。

**注意**

**debuginfo-install** 由 **yum-utils** 软件包提供。

2. 要验证是否安装了 debug 符号，请输入以下命令：

```
$ gdb which java

Reading symbols from /usr/bin/java...Reading symbols from /usr/lib/debug/usr/lib/jvm/java-21-version/bin/java-21-version.x86_64.debug...done.
(gdb)
```

## 3.2. 检查调试符号的安装位置

此流程解释了如何查找调试符号的位置。

**注意**

如果安装了 **debuginfo** 软件包，但您无法获取软件包的安装位置，请检查是否已安装了正确的软件包和 java 版本。确认版本后，再次检查调试符号的位置。

**先决条件**

- 在本地 sytem 上安装了 **gdb** 软件包。
  - 您可以在 CLI 上发出 **sudo yum install gdb** 命令，以在本地系统上安装此软件包。

- 安装了调试符号软件包。请参阅 安装调试符号。

**流程**

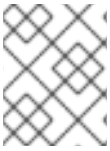
1. 要查找调试符号的位置，请使用 **gdb** 与 **java** 命令：

```
$ gdb which java

Reading symbols from /usr/bin/java...Reading symbols from /usr/lib/debug/usr/lib/jvm/java-21-openjdk-version/bin/java-version.x86_64.debug...done.
(gdb)
```

2. 使用以下命令浏览 ***-debug** 目录来查看库的所有调试版本，其中包括 **java**、**javac** 和 **javah** ：

```
$ cd /usr/lib/debug/lib/jvm/java-21-openjdk-version
```

```
$ tree

OJDK 21 version:
└── java-21-openjdk-version
├── bin
│   ...
│ ├──  java-java-version.x86_64.debug
│   ├── javac-java-version.x86_64.debug
│   ├── javadoc-java-version.x86_64.debug
│   ...
└── lib
├── jexec-java-version.x86_64.debug
├── jli
│   └── libjli.so-java-version.x86_64.debug
├── jspawnhelper-java-version.x86_64.debug
│   ...
```



**注意**

**javac** 和 **javah** 工具由 **java-21-openjdk-devel** 软件包提供。您可以使用 命令来安装软件包 **：$ sudo debuginfo-install java-21-openjdk-devel**。

## 3.3. 检查调试符号的配置

您可以检查和设置调试符号的配置。

- 输入以下命令获取安装的软件包列表：

```
$ sudo yum list installed | grep 'java-21-openjdk-debuginfo'
```

- 如果没有安装一些调试信息软件包，请输入以下命令安装缺少的软件包：

```
$ sudo yum debuginfo-install glibc-2.28-151.el8.x86_64 libgcc-8.4.1-1.el8.x86_64 libstdc++-8.4.1-1.el8.x86_64 sssd-client-2.4.0-9.el8.x86_64 zlib-1.2.11-17.el8.x86_64
```

- 如果要达到特定的断点，请运行以下命令：

```
$ gdb -ex 'handle SIGSEGV noprint nostop pass' -ex 'set breakpoint pending on' -ex 'break JavaCalls::call' -ex 'run' --args java ./HelloWorld
```

以上命令完成以下任务：

- 处理 SIGSEGV 错误，因为 JVM 使用 SEGV 进行堆栈溢出检查。
- 将待处理的断点设置为 **yes**。
- 在 **JavaCalls::call** 函数中调用 break 语句。在 HotSpot (libjvm.so)中启动应用程序的功能。

## 3.4. 在致命错误日志文件中配置调试符号

当 Java 应用因为 JVM 崩溃而停机时，会生成一个致命错误日志文件，例如： **hs_error**.**java_error**。这些错误日志文件在应用程序的当前工作目录中生成。crash 文件包含来自堆栈的信息。

**流程**

1. 您可以使用 **strip -g** 命令删除所有调试符号。
   以下代码显示了一个未经过的 **hs_error** 文件示例：

```
Native frames: (J=compiled Java code, j=interpreted, Vv=VM code, C=native code)
V  [libjvm.so+0xb83d2a]  Unsafe_SetLong+0xda
j  sun.misc.Unsafe.putLong(Ljava/lang/Object;JJ)V+0
j  Crash.main([Ljava/lang/String;)V+8
v  ~StubRoutines::call_stub
V  [libjvm.so+0x6c0e65]  JavaCalls::call_helper(JavaValue*, methodHandle*, JavaCallArguments*, Thread*)+0xc85
V  [libjvm.so+0x73cc0d]  jni_invoke_static(JNIEnv_*, JavaValue*, _jobject*, JNICallType, _jmethodID*, JNI_ArgumentPusher*, Thread*) [clone .constprop.1]+0x31d
V  [libjvm.so+0x73fd16]  jni_CallStaticVoidMethod+0x186
C  [libjli.so+0x48a2]  JavaMain+0x472
C  [libpthread.so.0+0x9432]  start_thread+0xe2
```

   以下代码显示了剥离的 **hs_error** 文件示例：

```
Stack: [0x00007ff7e1a44000,0x00007ff7e1b44000],  sp=0x00007ff7e1b42850,  free space=1018k
Native frames: (J=compiled Java code, j=interpreted, Vv=VM code, C=native code)
V  [libjvm.so+0xa7ecab]
j  sun.misc.Unsafe.putAddress(JJ)V+0
j  Crash.crash()V+5
j  Crash.main([Ljava/lang/String;)V+0
v  ~StubRoutines::call_stub
V  [libjvm.so+0x67133a]
V  [libjvm.so+0x682bca]
V  [libjvm.so+0x6968b6]
C  [libjli.so+0x3989]
C  [libpthread.so.0+0x7dd5]  start_thread+0xc5
```

2. 输入以下命令检查您是否有相同的调试符号版本和严重错误日志文件：

```
$ java -version
```

**注意**

您还可以使用 **sudo update-alternatives --config 'java'** 完成此检查。

3. 使用 **nm** 命令确保 **libjvm.so** 具有 ELF 数据和文本符号：

```
$ nm /usr/lib/debug/usr/lib/jvm/java-21-
openjdk-version/lib/server/libjvm.so-version.x86_64.debug
```

**其他资源**

- 在没有安装 debug 符号的情况下，崩溃文件 **hs_error** 不完整。如需更多信息，请参阅 因为 JVM 崩溃而关闭 Java 应用程序。

# 第 4 章 在 RED HAT ENTERPRISE LINUX 上更新红帽构建的 OPENJDK 21

以下小节提供了在 Red Hat Enterprise Linux 上更新红帽构建的 OpenJDK 21 的说明。

## 4.1. 使用 YUM 更新 RHEL 上的红帽构建的 OPENJDK 21

可使用 **yum** 系统软件包管理器更新已安装的红帽构建的 OpenJDK 软件包。

**先决条件**

- 必须具有系统上的 root 权限。

**流程**

1. 检查当前版本的 OpenJDK 版本：

```
$ sudo yum list installed "java*"
```

   此时会显示已安装的红帽构建的 OpenJDK 软件包列表。

2. 更新特定软件包。例如：

```
$ sudo yum update java-21-openjdk
```

3. 通过检查当前红帽构建的 OpenJDK 版本来验证更新是否正常工作：

```
$ java -version
```

**注意**

您可以在本地系统中安装 Red Hat build of OpenJDK 的多个主版本。如果您需要从一个主版本切换到另一个主版本，请在命令行界面(CLI)中运行以下命令，然后按照屏幕提示操作：

```
$ sudo update-alternatives --config 'java'
```

## 4.2. 使用存档更新 RHEL 上的红帽构建的 OPENJDK 21

您可以使用存档更新红帽构建的 OpenJDK。如果红帽构建的 OpenJDK 管理员没有 root 权限，这很有用。

**先决条件**

- 知道指向 JDK 或 JRE 安装的通用路径。例如： **~/jdks/java-21**

**流程**

1. 删除 JDK 或 JRE 的通用路径的现有符号链接。
   例如：

```
$ unlink ~/jdks/java-21
```

2. 在安装位置中安装 JDK 或 JRE 的最新版本。

**其他资源**

- 有关安装 JRE 的说明，请参阅使用存档在 RHEL 上安装 JRE 。
- 有关安装 JDK 的说明，请参阅使用存档在 RHEL 8 上安装红帽构建的 OpenJDK 。

*更新于 2024-05-10*